1) Interfacing simulators or their instructor consoles to LMSs.

Summary

Allow simulators or instructor/operator stations of simulators to retrieve information from the LMS and send tracking information of simulation events such as learner scores to the LMS. This need applies to software-based simulations as well as hardware-based simulators.

Requirements/Needs

Training events in simulations need to be scheduled and tracked in a Training/Learning Management System (T/LMS). Mastery in a simulator is often more indicative of readiness than mastery of web-based test questions. Many simulators have such fidelity that a certain amount of time in the simulator is required for certifying the learner. To optimize the learning experience in the simulation, the proper scenario needs to be scheduled along with the learning event that uses the simulation. Basic tracking information such as how long the learner was in the simulation, what was the scenario, were there any critical mistakes need to be sent from the training device to learning tracking system. Ideally, detailed tracking information should be sent to the management to verify skills and identify trends.

"Unique" requirements: When one who is familiar with SCORM and AICC thinks about scheduling and tracking, they think of a SCO or Assignable Unit (AU). However, hardware-based simulators or large-scale training events are not "content objects." Most of the specifications used in SCORM & AICC were developed with this in mind and language was carefully chosen to, for example, call the leaf nodes of the activity tree "learning events." However, the complete package applies other restraints requiring, for example, that a SCO be a URL-launchable asset that can find and use the ECMAscript-based API. Some creativity as been used to create SCORM-Compliant SCOs that can find a way to communicate to a particular training device. However, most of these creative methods can not be generalized and often require other specific software and hardware to be installed and configured manually for the particular situation.

Recommendations

There were at least three different approaches taken that had limited success. One approach to had a SCO create an initialization file for the simulator based on information that the SCO had access to such as learner name, completion status and mastery level of learning objectives, etc. This file was read by the simulator or the simulator IOS to initialize the simulator for the training event. Results of the simulation event were recorded to a file by either the simulator, the IOS, a performance assessment tool, instructor grading sheet tool, or an after-action debrief tool. This file was often formatted as XML. When the learner logged back into the LMS, the same SCO or that wrote out the initialization file or another SCO parses the file with the simulation results and sends information to the LMS. If this approach is used, both of the files need to be standardized. For example, the initialization file could be derived on the Military Scenario Definition Language (MSDL) and the results file should probably be formatted to the CMI XML standard. This approach also requires that all systems know where the files will be stored.

Another approach used a Java applet and/or ActiveX component in a SCO to communicate with the simulator – often continuously in real-time. Some of these approaches leveraged IEEE simulation interoperability standards such as the High Level Architecture (HLA) or Distributed Interactive Simulation (DIS) to provide the mechanisms for the communication between the applet or component and the simulator. One of the problems noted by these efforts was that the simulations often had to be altered in order to provide additional data for analysis. For example, a typical DIS or HLA simulator for military Distributed Mission Operations (DMO) training events only publishes external observable information that is in the DIS or the RPR Federation Object Model (FOM), and for some training events, that is not sufficient to adequately assess the learner in the simulator. Another problem with this approach is that a run-time infrastructure (RTI) is required to administer the data flow between the simulations and the applet or component in the SCO.

A third approach modified the sample runtime engine (RTE) to hand the learner off to a simulation training server when a particular asset type has been selected as the current activity in the activity tree. The RTE was also modified to provide a web-services interface for the simulation server to retrieve learner information and to send tracking information such as performance scores or learner mastery levels to the RTE. While this approach raises technical, philosophical, and security issues, it does provide for other possibilities such as competency modeling systems and standardized reporting and analysis tools and solves some of the problems in the current SCORM.

Brandt W. Dargue, ATF
Boeing Associate Technical Fellow
Training Technologies
Phantom Works Support Technologies - Logistics (St. Louis)
(314) 232-3165
(314) 650-7714 mobile
mailto:Brandt.W.Dargue@Boeing.com